How to automate Microsoft Excel from Visual Basic .NET

This article was previously published under Q301982

Article ID : 301982
Last Review : September 19, 2005
Revision : 8.0

SUMMARY

This article demonstrates how to create an Automation client for Microsoft Excel by using Microsoft Visual Basic .NET.

MORE INFORMATION

Automation is a process that allows applications that are written in languages such as Visual Basic to programmatically control other applications. Automation to Excel allows you to perform actions such as creating a new workbook, adding data to the workbook, or creating charts. With Excel and other Microsoft Office applications, virtually all of the actions that you can perform manually through the user interface can also be performed programmatically by using Automation.

Excel exposes this programmatic functionality through an object model. The object model is a collection of classes and methods that serve as counterparts to the logical components of Excel. For example, there is an **Application** object, a **Workbook** object, and a **Worksheet** object, each of which contain the functionality of those components of Excel. To access the object model from Visual Basic .NET, you can set a project reference to the type library.

This article demonstrates how to set the proper project reference to the Excel type library for Visual Basic .NET and provides sample code to automate Excel.

Create an automation client for Microsoft Excel

1. Start Microsoft Visual Studio .NET.
2. On the **File** menu, click **New**, and then click **Project**. Select **Windows Application** from the Visual Basic Project types. Form1 is created by default.
3. Add a reference to **Microsoft Excel Object Library**. To do this, follow these steps:
   a. On the **Project** menu, click **Add Reference**.
   b. On the **COM** tab, locate **Microsoft Excel Object Library**, and then click **Select**.

      Note Microsoft Office 2003 includes Primary Interop Assemblies (PIAs). Microsoft Office XP does not include PIAs, but they can be downloaded. For more information about Office XP PIAs, click the following article number to view the article in the Microsoft Knowledge Base:

      328912 (http://support.microsoft.com/kb/328912/) Microsoft Office XP primary interop assemblies (PIAs) are available for download

   c. Click **OK** in the **Add References** dialog box to accept your selections.
4. On the **View** menu, select **Toolbox** to display the Toolbox, and then add a button to Form1.
5. Double-click **Button1**. The code window for the form appears.
6. In the code window, locate the following code:

```
Private Sub Button1_Click(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles Button1.Click

End Sub
```

   Replace the previous code with the following code:

```
Private Sub Button1_Click(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles Button1.Click
    Dim oXL As Excel.Application
    Dim oWB As Excel.Workbook
    Dim oSheet As Excel.Worksheet
    Dim oRng As Excel.Range

    ' Start Excel and get Application object.
    oXL = CreateObject("Excel.Application")
    oXL.Visible = True
```

```
        ' Get a new workbook.
        oWB = oXL.Workbooks.Add
        oSheet = oWB.ActiveSheet

        ' Add table headers going cell by cell.
        oSheet.Cells(1, 1).Value = "First Name"
        oSheet.Cells(1, 2).Value = "Last Name"
        oSheet.Cells(1, 3).Value = "Full Name"
        oSheet.Cells(1, 4).Value = "Salary"

        ' Format A1:D1 as bold, vertical alignment = center.
        With oSheet.Range("A1", "D1")
            .Font.Bold = True
            .VerticalAlignment = Excel.XlVAlign.xlVAlignCenter
        End With

        ' Create an array to set multiple values at once.
        Dim saNames(5, 2) As String
        saNames(0, 0) = "John"
        saNames(0, 1) = "Smith"
        saNames(1, 0) = "Tom"
        saNames(1, 1) = "Brown"
        saNames(2, 0) = "Sue"
        saNames(2, 1) = "Thomas"
        saNames(3, 0) = "Jane"

        saNames(3, 1) = "Jones"
        saNames(4, 0) = "Adam"
        saNames(4, 1) = "Johnson"

        ' Fill A2:B6 with an array of values (First and Last Names).
        oSheet.Range("A2", "B6").Value = saNames

        ' Fill C2:C6 with a relative formula (=A2 & " " & B2).
        oRng = oSheet.Range("C2", "C6")
        oRng.Formula = "=A2 & "" "" & B2"

        ' Fill D2:D6 with a formula(=RAND()*100000) and apply format.
        oRng = oSheet.Range("D2", "D6")
        oRng.Formula = "=RAND()*100000"
        oRng.NumberFormat = "$0.00"

        ' AutoFit columns A:D.
        oRng = oSheet.Range("A1", "D1")
        oRng.EntireColumn.AutoFit()

        ' Manipulate a variable number of columns for Quarterly Sales Data.
        Call DisplayQuarterlySales(oSheet)

        ' Make sure Excel is visible and give the user control
        ' of Excel's lifetime.
        oXL.Visible = True
        oXL.UserControl = True

        ' Make sure that you release object references.
        oRng = Nothing
        oSheet = Nothing
        oWB = Nothing
        oXL.Quit()
        oXL = Nothing

        Exit Sub
Err_Handler:
        MsgBox(Err.Description, vbCritical, "Error: " & Err.Number)
    End Sub

    Private Sub DisplayQuarterlySales(ByVal oWS As Excel.Worksheet)
        Dim oResizeRange As Excel.Range
        Dim oChart As Excel.Chart
```

```
        Dim oSeries As Excel.Series
        Dim iNumQtrs As Integer
        Dim sMsg As String
        Dim iRet As Integer


        ' Determine how many quarters to display data for.
        For iNumQtrs = 4 To 2 Step -1
            sMsg = "Enter sales data for" & Str(iNumQtrs) & " quarter(s)?"
            iRet = MsgBox(sMsg, vbYesNo Or vbQuestion _
               Or vbMsgBoxSetForeground, "Quarterly Sales")
            If iRet = vbYes Then Exit For
        Next iNumQtrs

        ' Starting at E1, fill headers for the number of columns selected.
        oResizeRange = oWS.Range("E1", "E1").Resize(ColumnSize:=iNumQtrs)
        oResizeRange.Formula = "=""Q"" & COLUMN()-4 & CHAR(10) & ""Sales"""

        ' Change the Orientation and WrapText properties for the headers.
        oResizeRange.Orientation = 38
        oResizeRange.WrapText = True

        ' Fill the interior color of the headers.
        oResizeRange.Interior.ColorIndex = 36

        ' Fill the columns with a formula and apply a number format.
        oResizeRange = oWS.Range("E2", "E6").Resize(ColumnSize:=iNumQtrs)
        oResizeRange.Formula = "=RAND()*100"
        oResizeRange.NumberFormat = "$0.00"

        ' Apply borders to the Sales data and headers.
        oResizeRange = oWS.Range("E1", "E6").Resize(ColumnSize:=iNumQtrs)
        oResizeRange.Borders.Weight = Excel.XlBorderWeight.xlThin

        ' Add a Totals formula for the sales data and apply a border.
        oResizeRange = oWS.Range("E8", "E8").Resize(ColumnSize:=iNumQtrs)
        oResizeRange.Formula = "=SUM(E2:E6)"
        With oResizeRange.Borders(Excel.XlBordersIndex.xlEdgeBottom)
            .LineStyle = Excel.XlLineStyle.xlDouble
            .Weight = Excel.XlBorderWeight.xlThick
        End With

        ' Add a Chart for the selected data.
        oResizeRange = oWS.Range("E2:E6").Resize(ColumnSize:=iNumQtrs)
        oChart = oWS.Parent.Charts.Add
        With oChart
            .ChartWizard(oResizeRange, Excel.XlChartType.xl3DColumn, ,
Excel.XlRowCol.xlColumns)
            oSeries = .SeriesCollection(1)
            oSeries.XValues = oWS.Range("A2", "A6")
            For iRet = 1 To iNumQtrs
                .SeriesCollection(iRet).Name = "=""Q" & Str(iRet) & """"
            Next iRet
            .Location(Excel.XlChartLocation.xlLocationAsObject, oWS.Name)
        End With

        ' Move the chart so as not to cover your data.
        With oWS.Shapes.Item("Chart 1")
            .Top = oWS.Rows(10).Top
            .Left = oWS.Columns(2).Left
        End With

        ' Free any references.
        oChart = Nothing
        oResizeRange = Nothing
    End Sub
```

7. Add the following code to the top of Form1.vb:

```
Imports Microsoft.Office.Core
```

Test the automation client

1. Press F5 to build and to run the program.
2. On the form, click **Button1**. The program starts Excel and populates data on a new worksheet.
3. When you are prompted to enter quarterly sales data, click **Yes**. A chart that is linked to quarterly data is added to the worksheet.

REFERENCES

For more information, visit the following Microsoft Developer Network (MSDN) Web site:

Microsoft Office Development with Visual Studio
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnoxpta/html/vsofficedev.asp (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnoxpta/html/vsofficedev.asp)

For more information about Excel and Visual Basic, click the following article number to view the article in the Microsoft Knowledge Base:

219151 (http://support.microsoft.com/kb/219151/) How to automate Microsoft Excel from Visual Basic

---

APPLIES TO

- Microsoft Visual Basic .NET 2003 Standard Edition
- Microsoft Visual Basic .NET 2002 Standard Edition
- Microsoft Office Excel 2003
- Microsoft Excel 2002 Standard Edition

Keywords: kbhowto kbautomation kbpia KB301982